Iwan Martin

# Optically Studying Spin Lifetimes in Wse2/Graphene Heterostructures

**Undergraduate Research Thesis**

Ohio State University

Autumn 2018

## Acknowledgements:

I'd like to acknowledge Professor Roland Kawakami for his mentorship and for providing me the opportunity to pursue this research. I would like to thank Professor Roberto Myers for sitting my oral defense committee and giving me feedback on my thesis. I would also like to acknowledge all the people in the Kawakami group for their assistance and mentoring, with special mention to Kelly Luo.

Above all I would especially like to give mention to Michael Newburger and Elizabeth McCormick for the large amount of effort they put into mentoring and working with me. I can honestly say I feel my experience in the lab taught me more than the entire sum of my undergraduate coursework combined, and that could not have happened without the work they put in.

# Table of Contents

# 1. Summary

In early 2017 until late 2018 I worked in the Kawakami group studying 2D materials and heterostructures using optical measurements. This document will discuss the work I did while working in the Kawakami group. It will discuss the experimental methodology and equipment, various contributions I made to the lab, and the $WSe_2$/Graphene study I assisted with in 2018.

The focus of this research was in studying spin dynamics for 2D materials. The work performed involved various supporting tasks and tool development as well as setting up and running experimental measurements within the lab.

## 1.1 Overview of Contributions

The primary contributions I had was in performing measurements and programming. In terms of measurements I was significantly responsible for taking measurements in $WSe_2$/Graphene devices for much of 2018 and assisted in some $WS_2$ device measurements in 2017. In terms of programming tasks, I redeveloped multiple programs for taking TRKR measurements, photocurrent measurements, and reflectivity measurements. I made improvements to how these programs were organized in terms of both visual interface and the underlying code for easier use and alteration. I also improved the way data was stored and organized. I made sure that the X, Y, and R

components were all stored, and I created images of the scans to easily be able to find the right data.

# 2. Optical Measurement Methodologies

The measurements I performed were primarily optical measurements and the methodology I will be discussing here is for the Photoluminescence (PL), Time Resolved Kerr Rotation (TRKR), and Photoconductivity (PC). I will also be discussing in extensive detail about Lock-In measurements with regards to optics.

Optical measurements have multiple advantages over the use of purely electrical measurements. Key advantages are the small spot size (<1μm), the ability to have picosecond timescale resolution, and the ability to create setups that use multiple lock-ins to isolate the desired signal. In this section I will discuss various aspects of the optical measurements that I performed or assisted with.

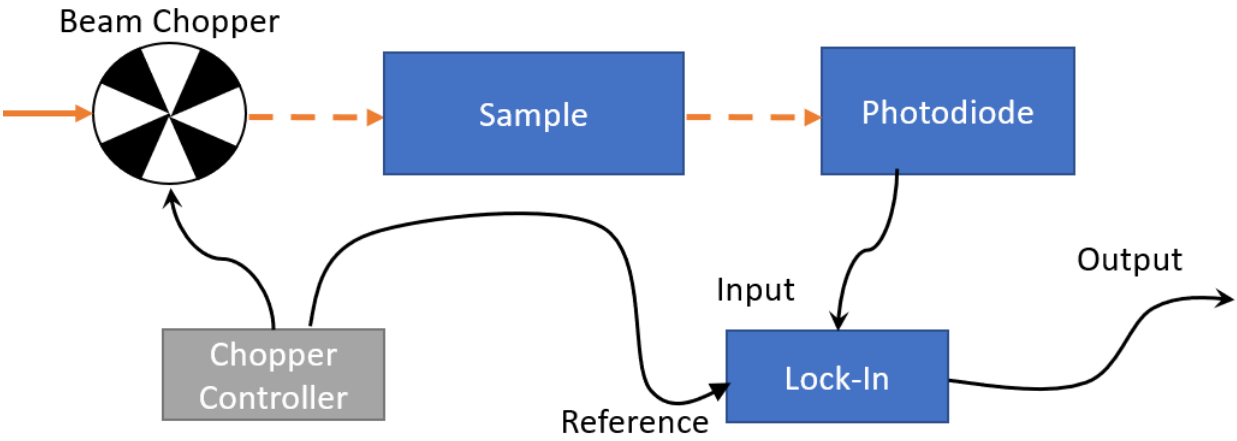
## 2.2 Lock-in Measurements

Key to many of the optical measurements performed within the Kawakami group's lab is lock-in based measurements. These devices function as a style of notch filter that isolates a frequency by exploiting the orthogonality of frequency components in Fourier space. The following equation describes the output signal in terms of its reference frequency, and input signal.

$f_{ref} = reference\ frequency$
$\varphi = phase\ offset$
$U_{in} = input\ signal$
$U_{out} = output\ signal$

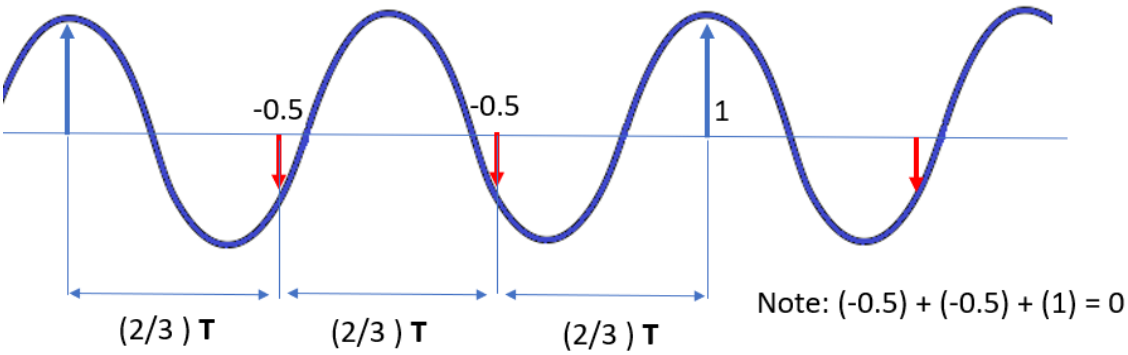
$$U_{out}(t) = \frac{1}{T}\int_{t-T}^{t} \sin\left[2\pi f_{ref} * s + \varphi\right] U_{in}(s)ds$$

*This equation describes the output signal in terms of the frequency components of the input signal*



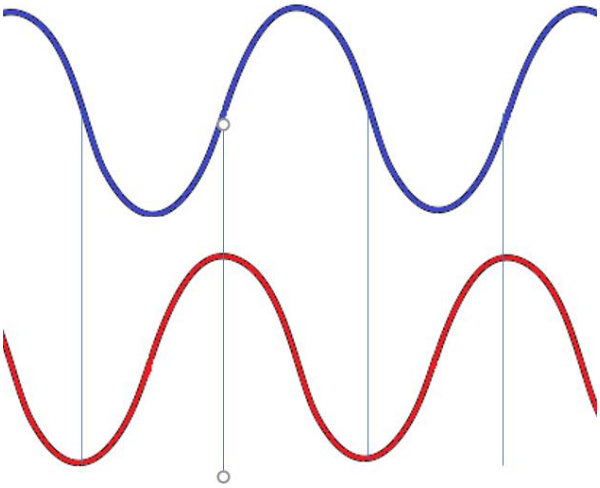

*Abstract diagram of a single lock-in setup where the sample under test is chopped by a beam chopper at a frequency before it interacts with the sample, then the effects are measured at that frequency by way of the lock-in filter.*

Essentially Lock-In's multiply a signal by a sinusoidal wave generated at the frequency of the reference frequency and generates a running average of the signal. For long averages any components not on the same frequency as the reference signal will cancel itself out due to the addition of a symmetric positive and negative energy components to the signal.

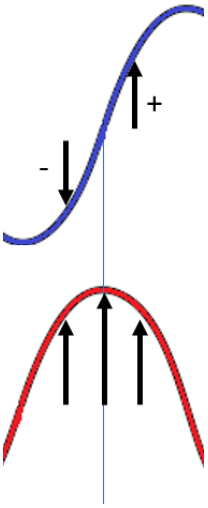


*Example of impulse train at 2/3$^{rds}$ the reference frequency used as an input signal, the multiplication of the input and reference frequency has 3 elements per period, these elements add to 0.*

In addition to frequency, phase is also an important aspect of the lock-in filtering process. Components that are 90 degrees out of phase with the generated sin wave will also zero out.

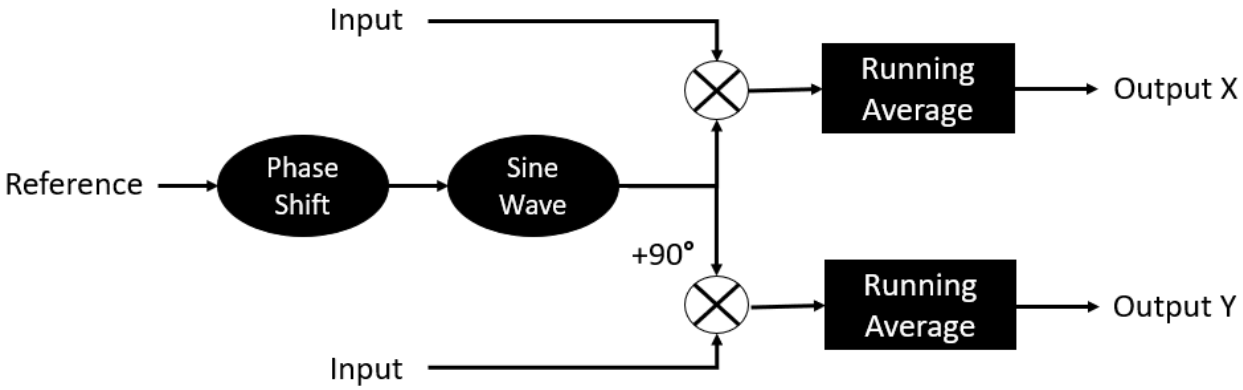


*Example of two sinusoids 90 degrees out of phase.*

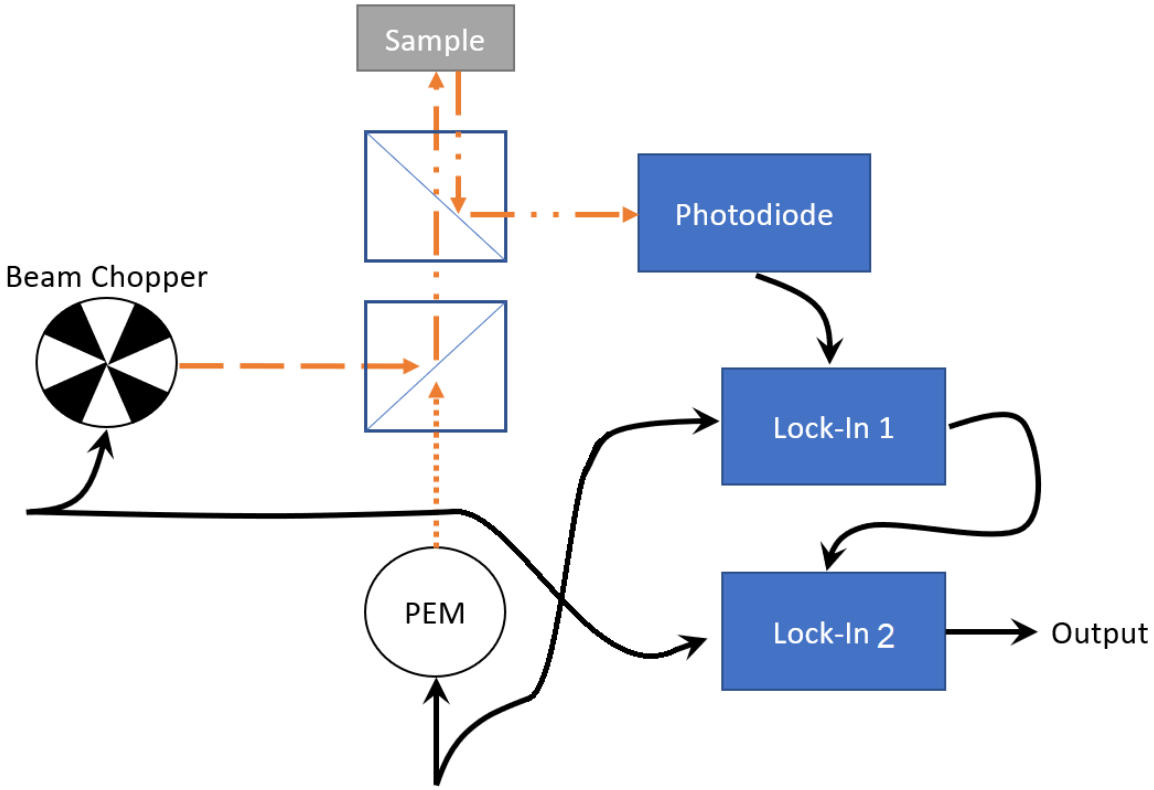



*Note: anti-symmetry of the node for the blue wave, and symmetry of the red wave leads to cancellation of energy when the two are multiplied and averaged*

The issue of phase is dealt with by adjusting the phase of the generated sine wave to match that of the measured effect, or by examining two sine waves, each themselves 90 degrees out of phase with each other. The total energy of the signal can be recovered by using the formula $R = \sqrt{X^2 + Y^2}$.




*Simplified block diagram for a lock-in device*




*Example of a full TRKR setup with 2 lock-in measurements. The second reference frequency is supplied by a Photo-Elastic Modulator (PEM).*

## 2.3 Waveplates & Photo-Elastic Modulators

Waveplates are optically transparent plates that experience anisotropy with relation to their permeability. Since this affects how fast light travels through a medium, light is slowed along one axis relative to another, this is typically described in terms of a “fast axis” and “slow axis”. The number of wavelengths of light that the slow axis has been slowed by in comparison to the fast axis is the wave number of a waveplate. The wave number will actually vary depending on the wavelength of light, typically waveplates are made to have ½ or ¼ wave number, the ½ waveplate will provide a flat rotation of the light without altering the helicity, the ¼ waveplate is

used to alter the helicity of a wave, a combination of the two can convert any possible variation of ellipticity into any other variation of ellipticity by rotating the plate to an appropriate position. A Photo-Elastic Modulator (PEM) is a waveplate that is modulated using the piezo-electric effect. This alters the wave number of the PEM according to the voltage applied, in the Kawakami group this is varied between either ½ wave or ¼ wave depending on needs. This is then modulated at a set frequency allowing for a frequency effect to be introduced to the measurement, by keying a lock-in to this frequency we can improve optical measurements.

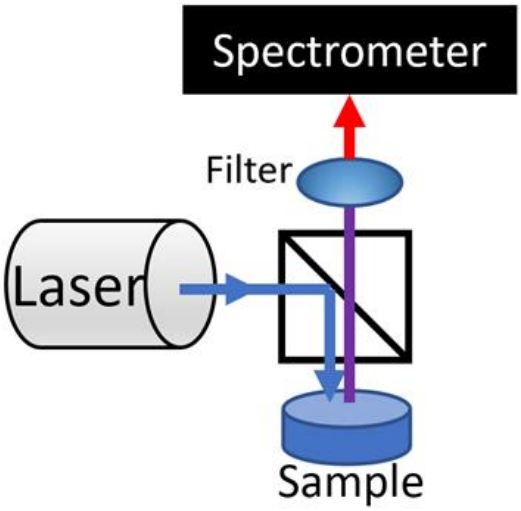
## 2.3 Beam Movement

In order to provide optical imaging, it's essential to be able to move the beam in relation to the device. Because of the difficulty of laterally moving a beam while maintaining the carefully achieved calibration of the beam, it's much easier to move the device underneath the beam. In the Kawakami group this is achieved through a set of Attocube linear positioners. These have both an open loop coarse movement and a closed loop fine movement mode of travel. The coarse movement relies on step motors for positioning, while the fine movement uses a piezo-electric scanner to position the device. Fine movement has extremely high precision movement well below the ~0.5μm possible resolution mandated by minimum spot size.

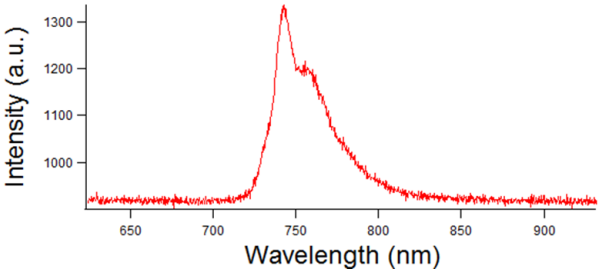
## 2.4 Photoluminescence

Photoluminescence (PL) allows for the band structure of a substance to be examined. These measurements involve exciting the electrons in a material using laser light, then measuring the wavelengths of the emitted light with a spectrometer. The photoluminescent light can be isolated by using a notch filter that removes the specific wavelength of the laser light. Different band gaps will create different peaks in the

spectroscopy graph, and by examining the size, shape, and position of these peaks, the band structure can be investigated.




*Simplified Photoluminescence Beamline Diagram*

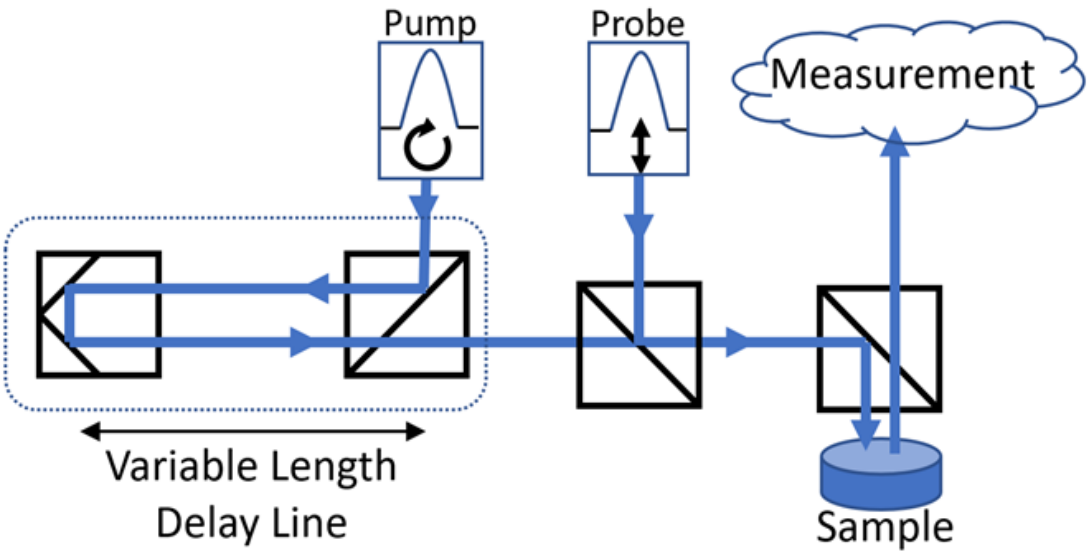



*Example of Photoluminescence curve, the different peaks observed in this curve correspond to bandgaps in the band structure.*

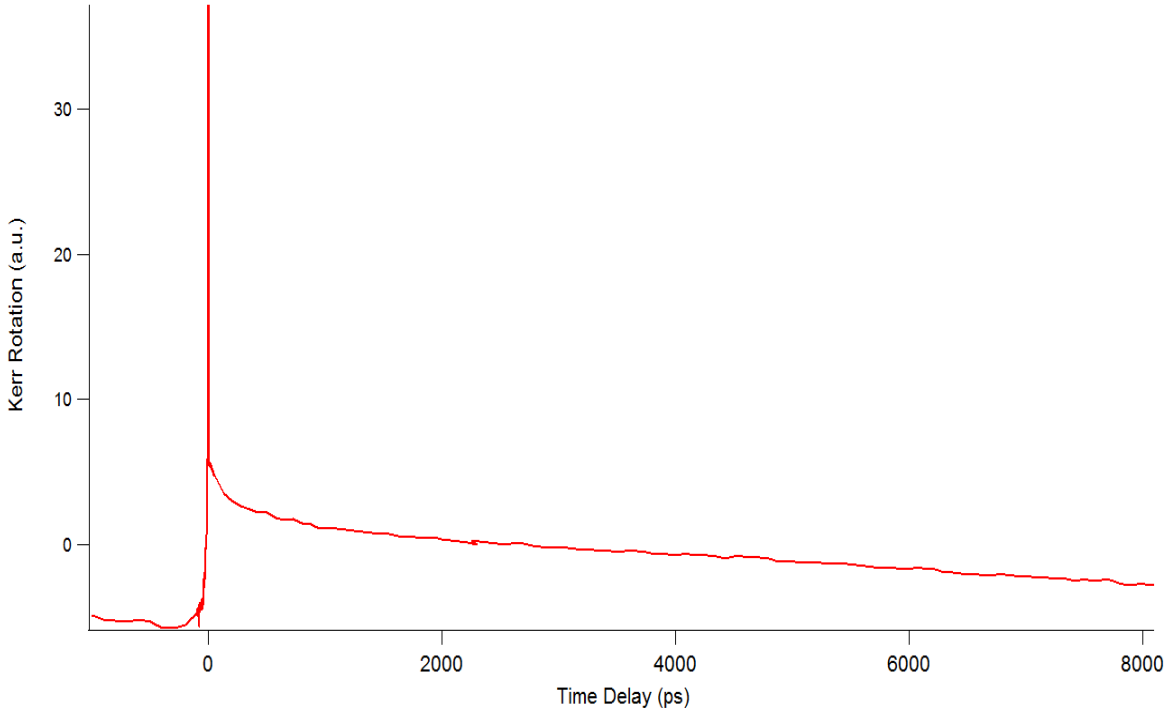
## 2.5 Kerr Rotation

Kerr Rotation (or the Kerr effect) is a slight rotation in the linear polarization of light that occurs when light reflects off a surface with spin polarity. In a perpendicular reflection (in the work discussed here we exclusively used perpendicular reflections) this is caused by anisotropic permittivity or permeability. Since light reflects slower along one axis than another the light experiences a rotation, linearly polarized light can become elliptically polarized when reflected. This anisotropy can be caused by spin alignment being in a preferential direction.

For TMD’s the spin structure can be manipulated using circularly polarized light. These two effects can be used in conjunction, with a pulse of circularly polarized light being used to create spin polarity in the material and a pulse of linearly polarized light being used to measure the spin polarity of the material. If we repeat this measurement with a variance in the difference in time (a time resolved measurement) we can measure the time it takes for the material to relax back to a neutral spin polarity (the spin lifetime). Time resolved measurements can be achieved by splitting the beam and sending one beam (either the measurement or spin injection beam) down a delay line, an arrangement where the distance between a retroreflector and a beam splitting cube is expanded or contracted to change the travel time of the beam. With the equipment in the Kawakami group lab, we can measure a delay of up to ~15ns, with a resolution limited by the pulse width (under best conditions ~0.5ps). Polarity is controlled either by using a quarter wave plate to induce circular polarity or using a Photo-Elastic Modulator (A device that functions as a waveplate that oscillates its wave number in tune with a reference frequency, moving from quarter waveplate in one axis to quarter waveplate along the orthogonal axis of the plate).



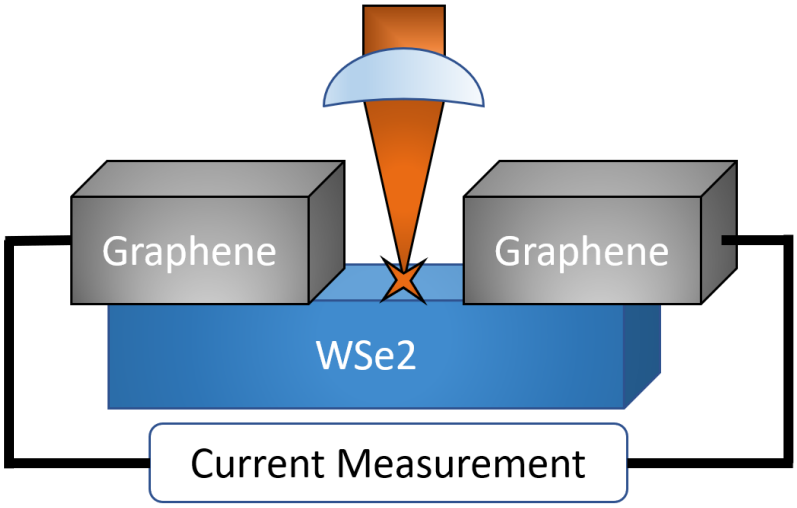

*Simplified TRKR Beamline Diagram A More Complex Diagram*




*Example of TRKR measurement, taken at 8K with 606nm laser, on $WS_2$ flake. Note in this case there is a negative noise offset in the system and a negative peak that happens before the primary peak due to exchange of measurement and excitation roles for the pump and probe beams*

## 2.6 Photoconductivity

Photoconductivity is when a material becomes more electrically conductive due to exposure to light, this occurs because of excitation of electrons into the conduction band. To measure this a voltage bias is applied across the device with a gate voltage also being applied to drive the TMD towards being P or N type. By measuring the change in current across the device we can determine the level of photoconductivity, by changing the position of the laser on the sample we determine how it changes on different parts of the device.




*Simplified diagram of photoconductivity measurements*

# 3. Scientific studies of $Wse_2$/Graphene

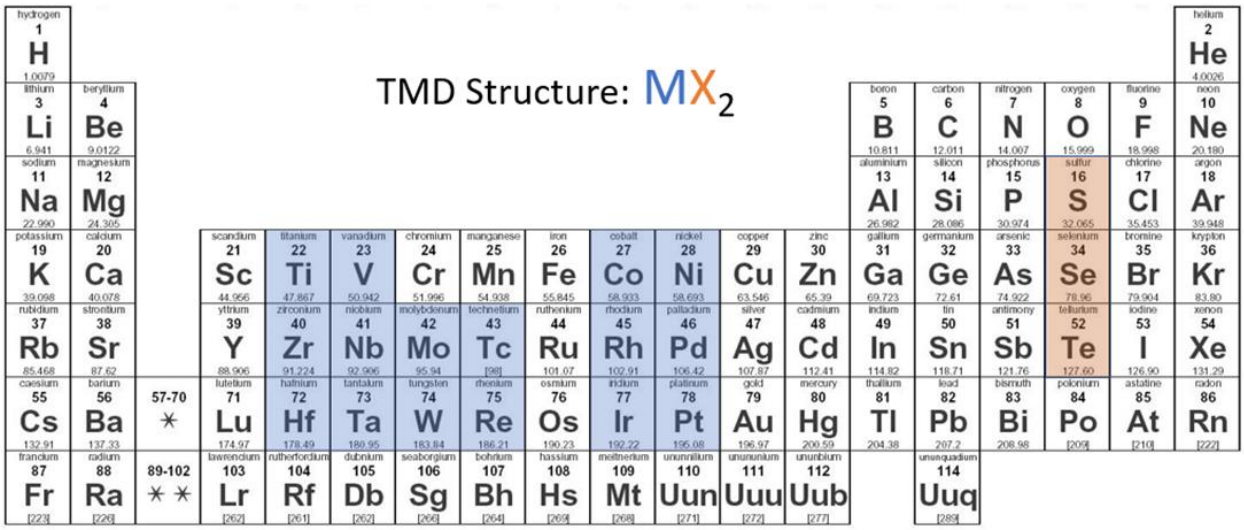
## 3.1 Overview of Spintronics

Spintronics is the field of study centered around creating novel computing devices that exploit the spin property of matter. Theory states that information carried and managed through magnetic fields and changes in spin direction can be much more energy efficient than information carried through the movement and accumulation of charge (as traditional silicon computing devices use today).

At a fundamental level a digital computing device needs three components: storage of information in format that allows reading and writing, transport for the movement of information between storage devices, and gating for the control of when information is transported. All of these components are important, the research work I have done has focused on learning more about the transport and gating of information through the examination of how a particular class of materials called Transition Metal Dichalcogenides(TMD's) function.

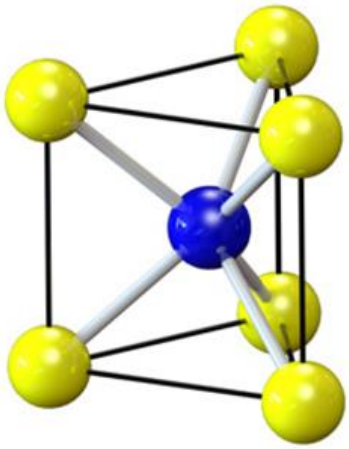
## 3.2 Transition Metal Dichalcogenides

A Transition Metal Dichalcogenide(TMD) is a 2D material formed of a transition metal layer bonding to two layers of chalcogenides.

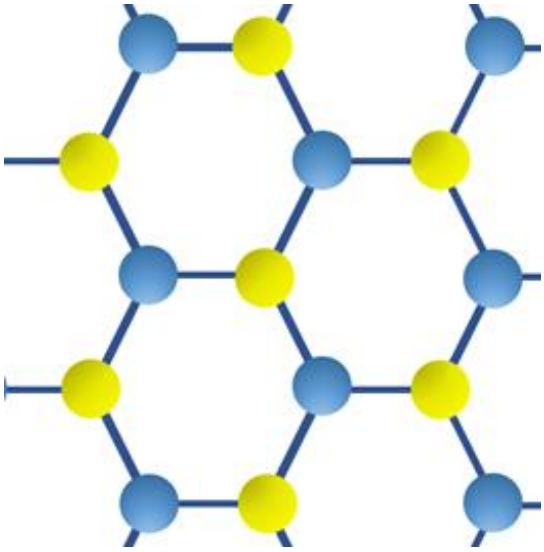



*Periodic Table with TMD elements highlighted, a TMD crystal is formed from a combination of the brown and blue elements in a two to one ratio.*

There are numerous possible combinations of these elements that will form a stable TMD's, however for this field of study we are primarily interested in $MoS_2$, $WS_2$, $MoSe_2$, $WSe_2$, $MoTe_2$.

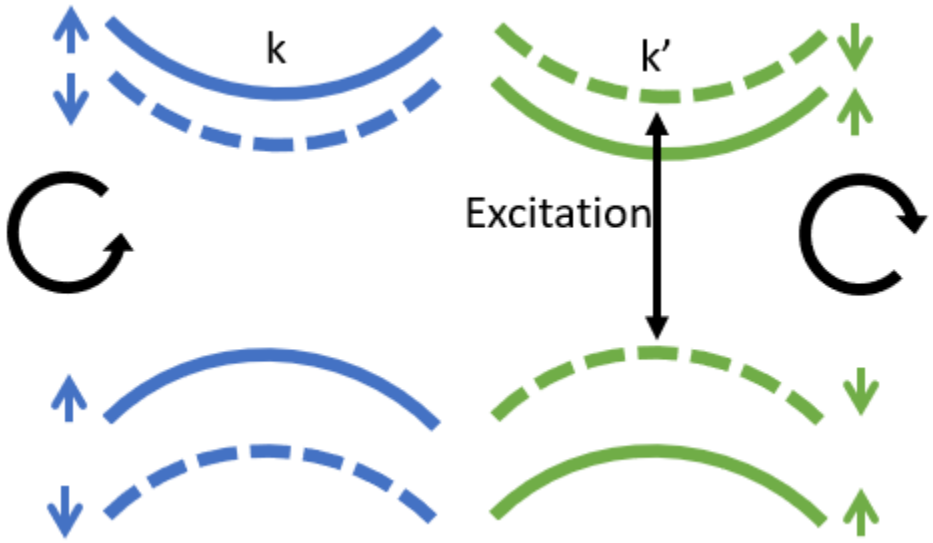


*TMD Unit Cell*
*in yellow: chalcogenides,*
*in blue: transition metals*



*Top View of TMD Structure*

Because these structure break inversion symmetry, there is a separation of energy for the K and K’ valleys. This makes them extremely useful for optical studies as it allows for selective excitation of the band structure, creating spin information.




*TMD Band structure diagram. A laser pulse of the appropriate wavelength and polarization will excite the electrons of a single spin without exciting the other electrons. Circularly polarized light will excite in only either the K or the K’ valley depending on left vs. right hand polarization.*

## 3.3 Graphene

Another material of importance for this study is graphene, a hexagonal 2D lattice of carbon atoms.

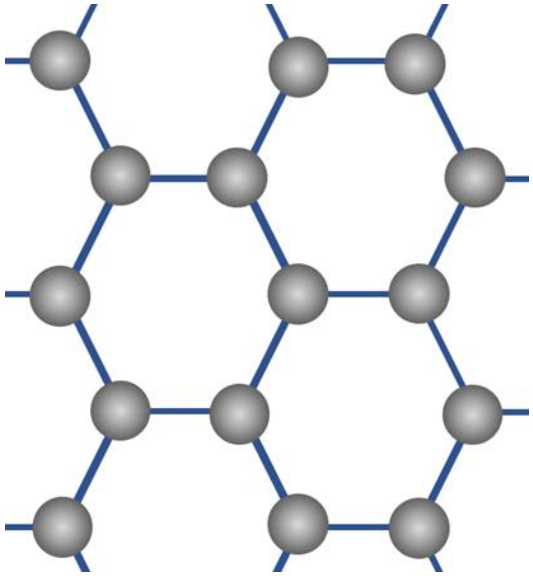


*Figure 5: Graphene lattice consisting of carbon atoms*

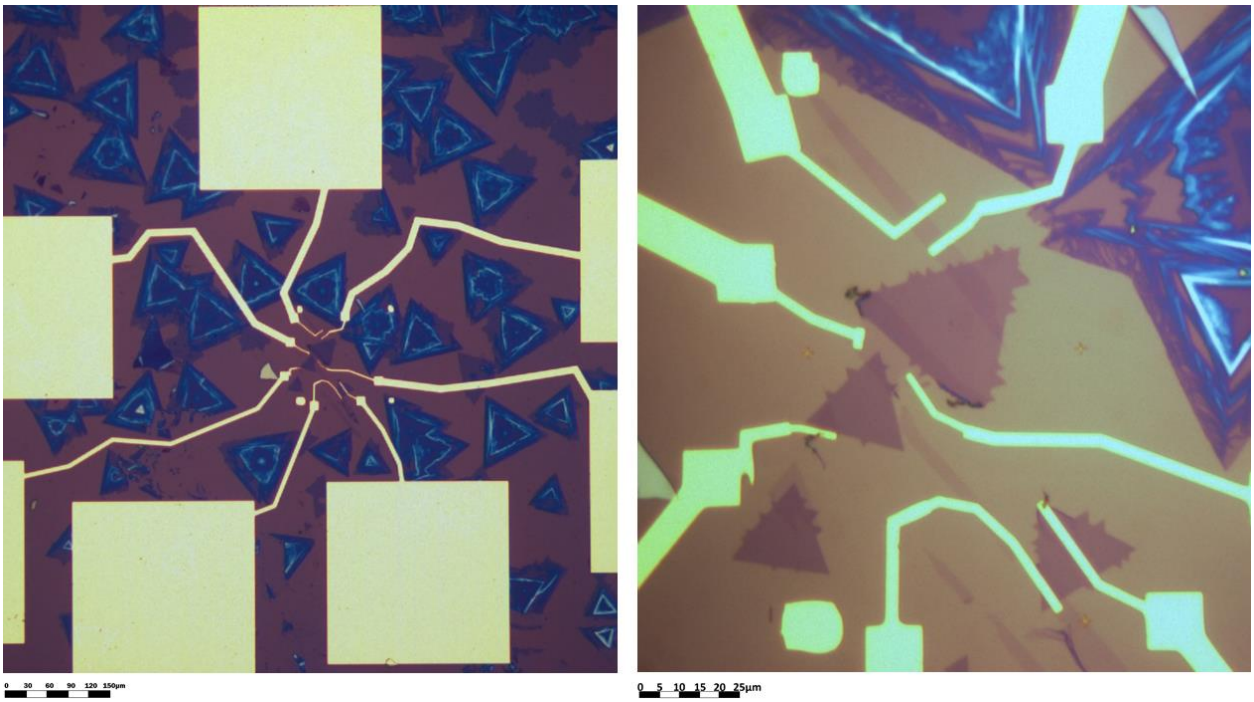
Graphene is known to have extremely high conductivity, it also demonstrates strong spin current properties [2] but lacks the ability to optically create spin information.

## 3.4 Experimental Setup

Since TMD's allow for optical control of spins and magnetic fields in general within a device. The interactions between TMD's and other materials with relation to their spin behavior is still not well understood. There is potential for a TMD flake to be used to create spin information then have the spin information travel into graphene. This would be useful because TMD's do not show good spin transport, and graphene does not allow for optical control. This study involved creating a heterostructure of $WSe_2$ and

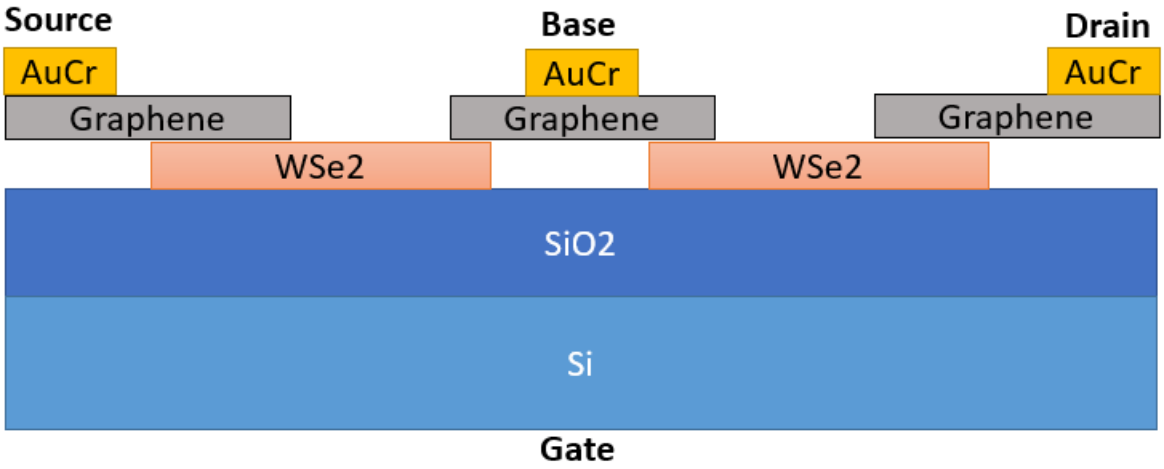
graphene with the intent of studying the spin dynamic interactions between the two materials.

The devices studied here were created by first creating a field of $WSe_2$ flakes upon a $SiO_2$ substrate using a chemical vapor deposition method. Viable single layer flakes were then located, and strips of graphene were transplanted onto the flakes. Chromium-gold contacts for electrical measurement were created using electron sputtering.
The contacts were wire bonded to allow for electrical measurement and to apply voltage biases. The two devices were created using these methods. They were studied using a combination of the described optical/electrical measurements under different applied voltage bias conditions.



*On the left microscope image of $WSe_2$ graphene device, on the right zoomed in image of the device. Bright yellow regions are chromium-gold contacts for electrical*

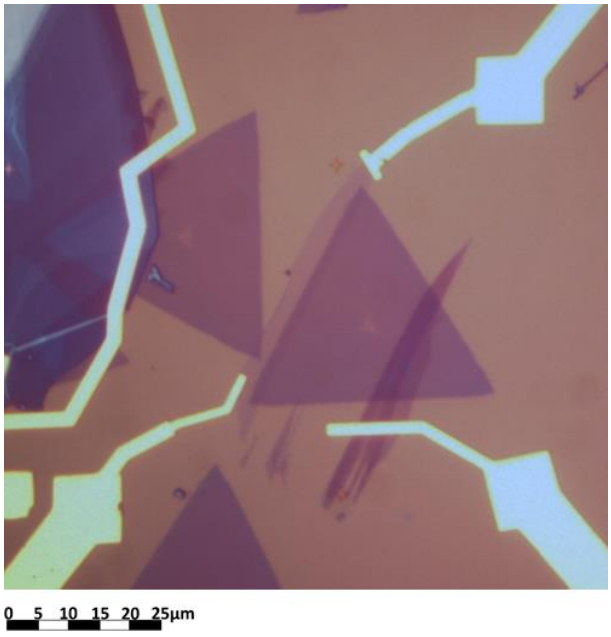
*measurement, the graphene is visible as a faint shadow that overlaps with the triangular $WSe_2$ flakes.*



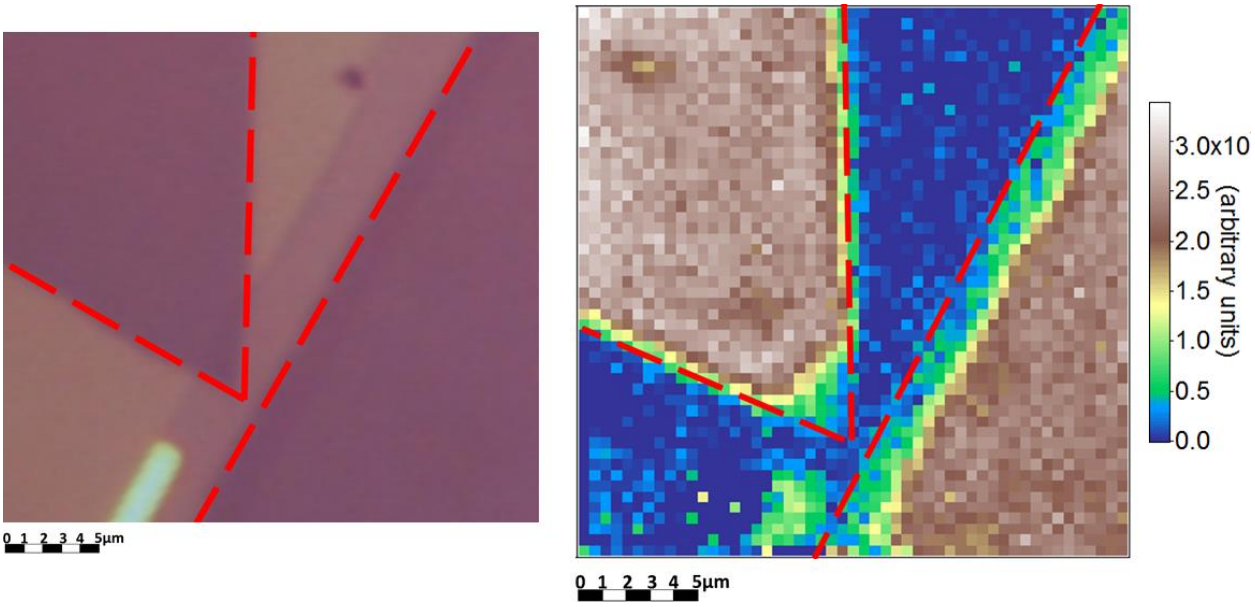

*Cross-section diagram showing construction of $WSe_2$/Graphene device. Measurements were performed with an applied bias between the gate (in the form of the silicon layer of the substrate), and the base (a metal contact placed onto the central graphene bridge).*

## 3.5 Experimental Results

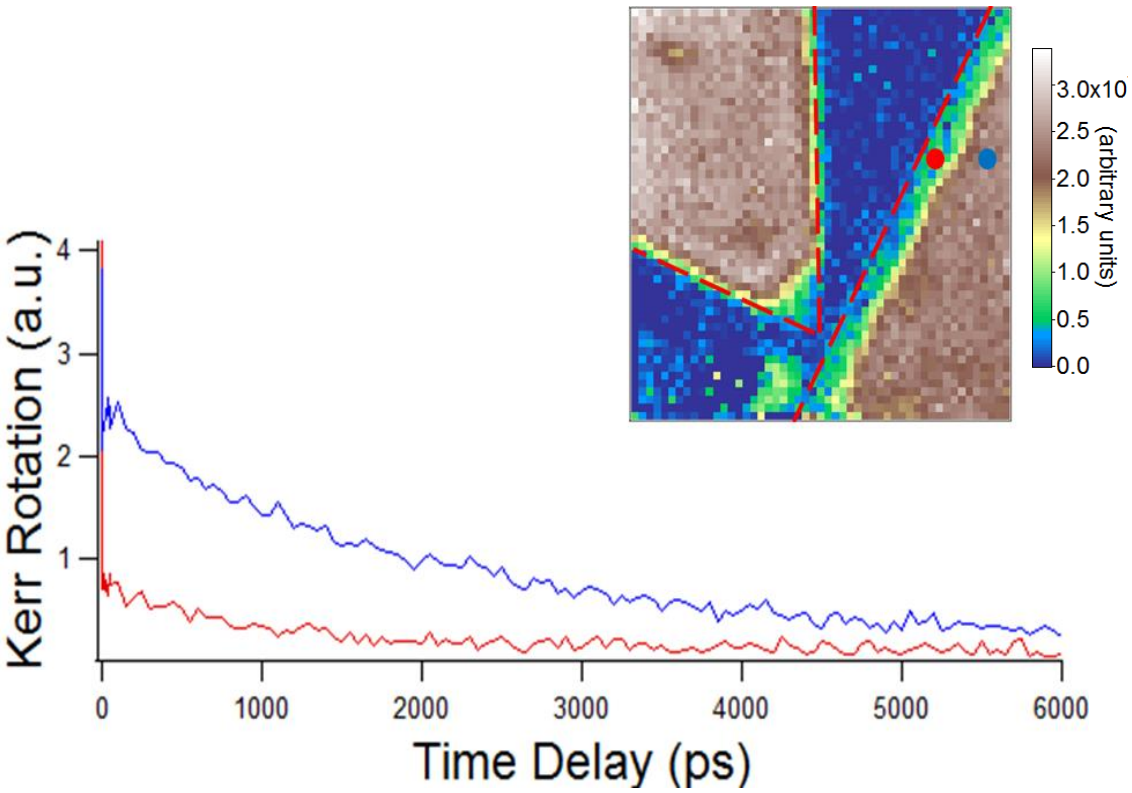
The key experiments we will discuss here are photoconductivity and Time Resolved Kerr Rotation, the following figures describe the results.



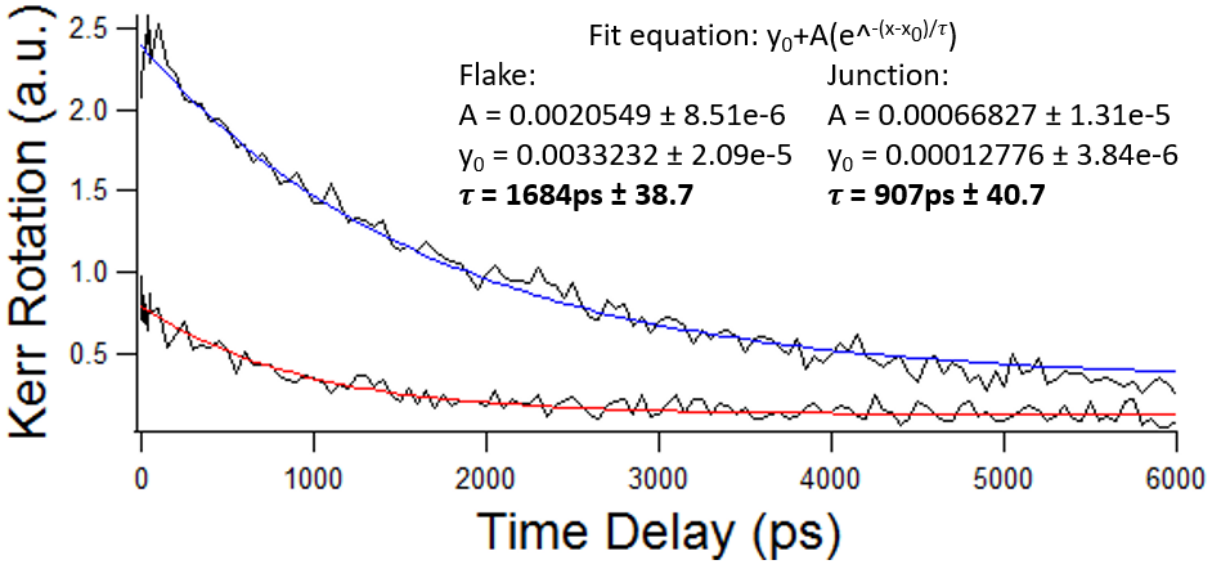

*Microscope images of second device, graphene overlap can be seen as a shadow over the flake and substrate.*



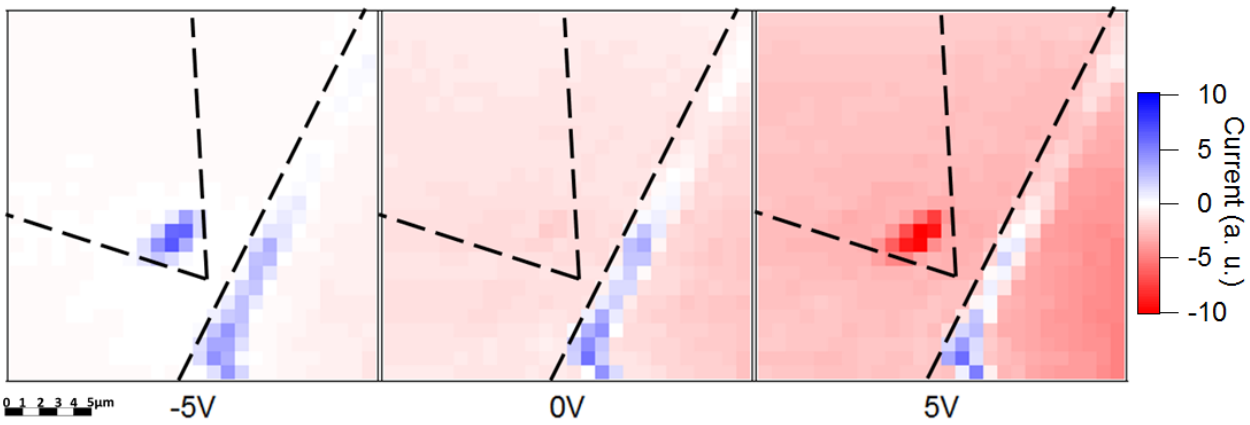

*TRKR map taken on device, shorter timelines can be observed in all regions where there is a graphene overlap. Map was taken at 8K with 734nm laser at 30v gate*




*TRKR measurement on junction region (red) and flake (blue). Measurements performed on second device. Copy of previous TRKR map with dots for locations is shown. Measurements taken with 734nm laser at 30v gate.*




*Measurement from previous figure with exponential decay fit applied. Junction region shows dramatically reduced lifetimes compared to flake region.*




*Photoconductivity maps at different voltage biases for second device, all maps were taken with a 30v gate voltage applied. The position of the two flakes is outlined in black, strongest photocurrent reactions were happening at the interface of graphene and $WSe_2$ with the most powerful reaction on the flake with smallest junction area.*

For these devices it was observed that TRKR lifetimes were dramatically shorter in the junction region than they were either upon the main body of the flake or in previous studies [1].

My specific contributions to this project was to run scans for photocurrent, PL, and TRKR. I also rewrote the programs that performed these measurements to be more user friendly and record X, Y, and R data in a method that is easily found. I also changed these programs to record an image of the graph allowing for easy viewing of data.

## 3.6 Conclusion

The graphene $WSe_2$ junction shows far reduced spin lifetimes when compared to the non-junction region. This suggests that the spin information is travelling into the graphene layer. These junction regions also show strong photoconductivity responses.

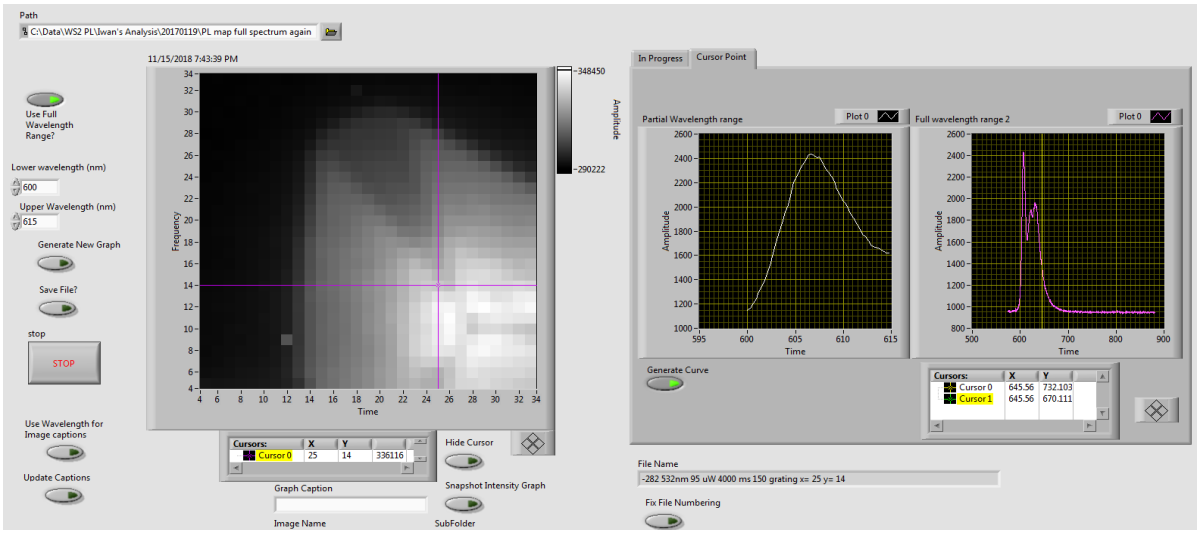
These two facts tied together suggests that spin information is travelling into the graphene through a movement of charges.

# 4. Lab Contributions:

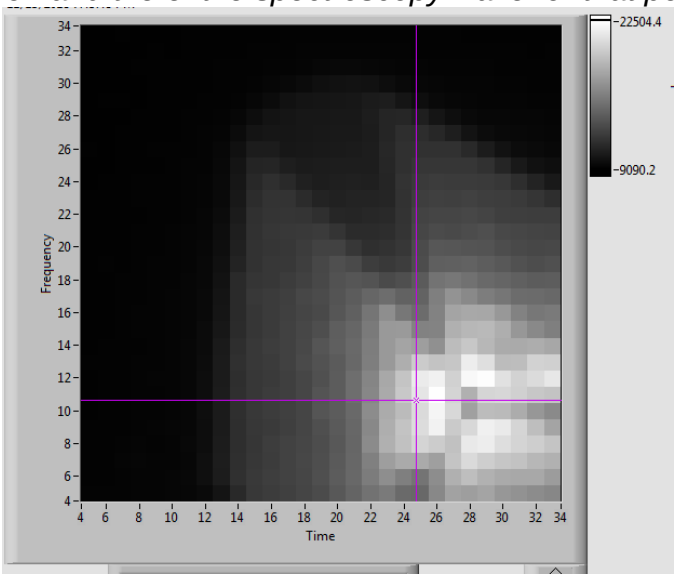
What follows is a summary of various projects and contributions I made during my time working at the lab. Primarily my contributions took the form of making improvements to existing experiments and Labview programs, but they also included the creation of some new tools.

## 4.1 PL map viewer

For the $WS_2$ project we needed to be able to easily sort through a point by point photoluminescence (PL) scan and try to quickly visualize where certain peaks were being seen. I accomplished this by writing a LabVIEW program where an image map was created with each point represented as the integration of the point's PL scan. The cursor could be moved over the individual points to view the PL scan for that point, and then the integration area could be changed for the map, allowing you to easily see where peaks were present in the map.




*Main interface of PL viewing and integrating program, on the left: map showing point by point integration, on the right: the select point's spectroscopy wave over the integrated region and the entire spectroscopy wave for that point.*



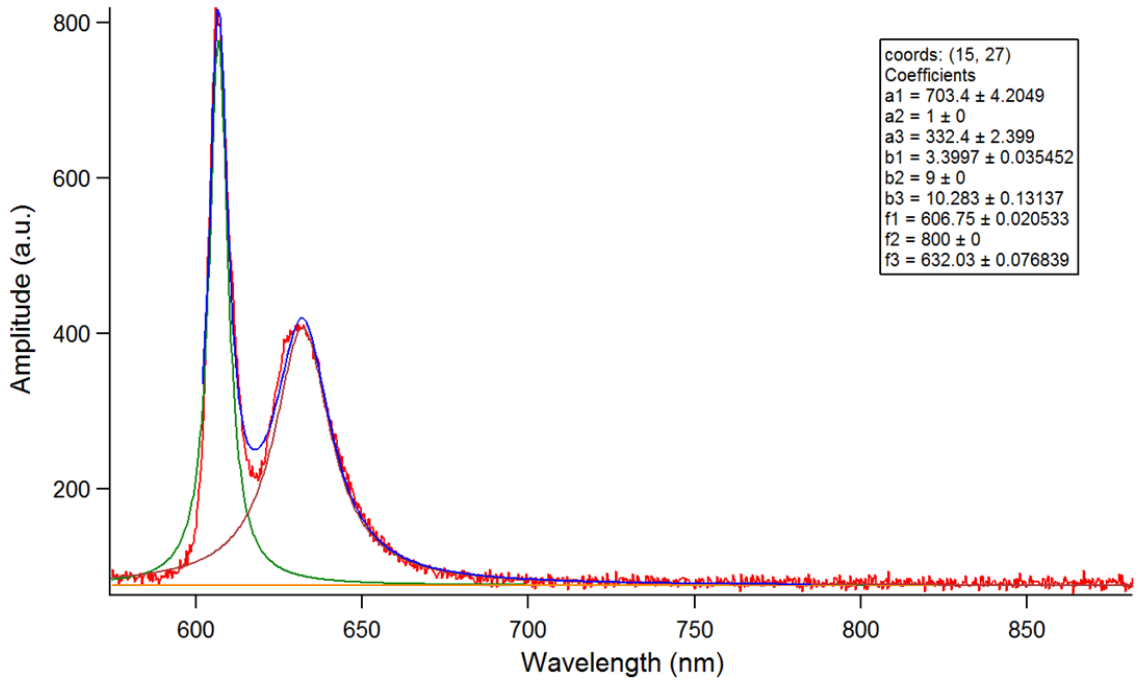

*This is an example of the same map integrated over a select range, note the increase in relative brightness around the triangular region suggesting the presence of a bandgap within those selected wavelengths*

## 4.2 Igor curve fitting program

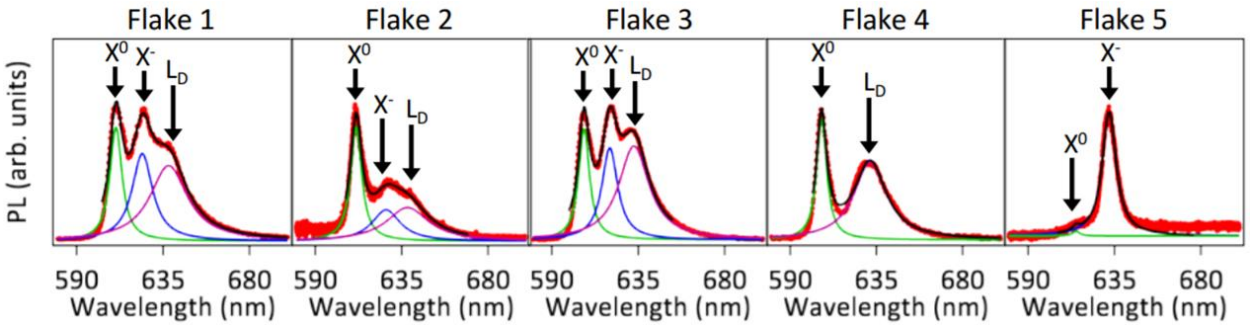
While the PL map viewing tool was useful for generating a quick analysis of

spectroscopy, the best way to determine the strength of multiple overlapping

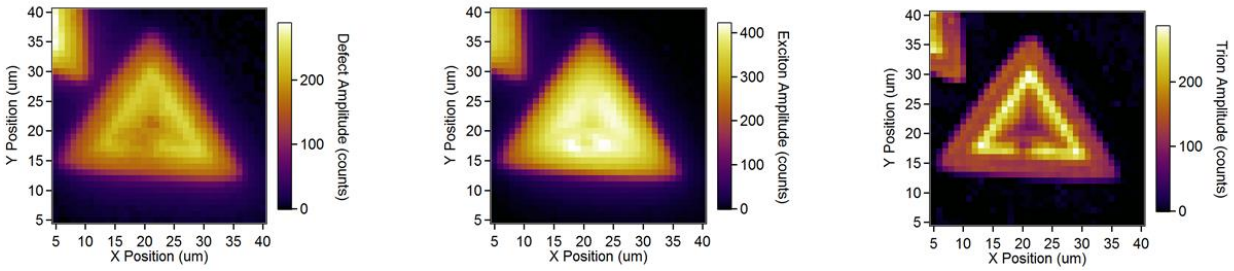
photoluminescence components is by curve fitting Lorentzian curves (something not easily done in LabVIEW). To do curve fitting on a point by point photoluminescence map I created a program written in Igor's C++ based script that would fit curves for each individual point and generate images representing each individual point. I determined that it would fit best when the things we expected to be constants across the map (e.g. wavelength of a peak) were actually treated as variables with narrow constrictions (e.g. wavelength is between 729.5nm and 730.5nm instead of being 730nm) In some cases the curve fitted maps showed strong improvement over the integration method when showing the presence of PL peaks on the sample.




*Example of curve fitting in action, the two green and brown waves combine to form the blue wave which is being fit to the data (in red), the amplitude coefficients can be used to form a map.*




*Examples of waves across different $WS_2$ flakes, Figure published in 2D materials [1]*

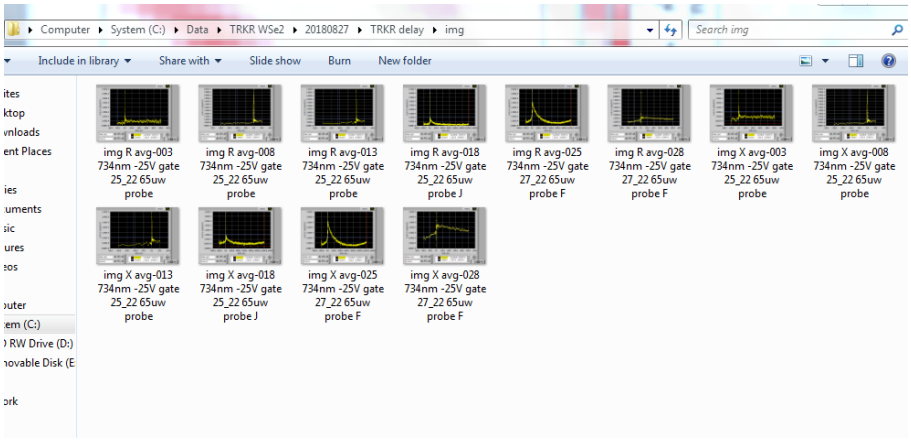



*From left to right maps correspond to Defect, Exciton, Trion peaks for a single $WS_2$ flake. Figure published in 2D Materials [1]*

Code for this project is including in the appendix [A1].

## 4.3 Taking images of charts

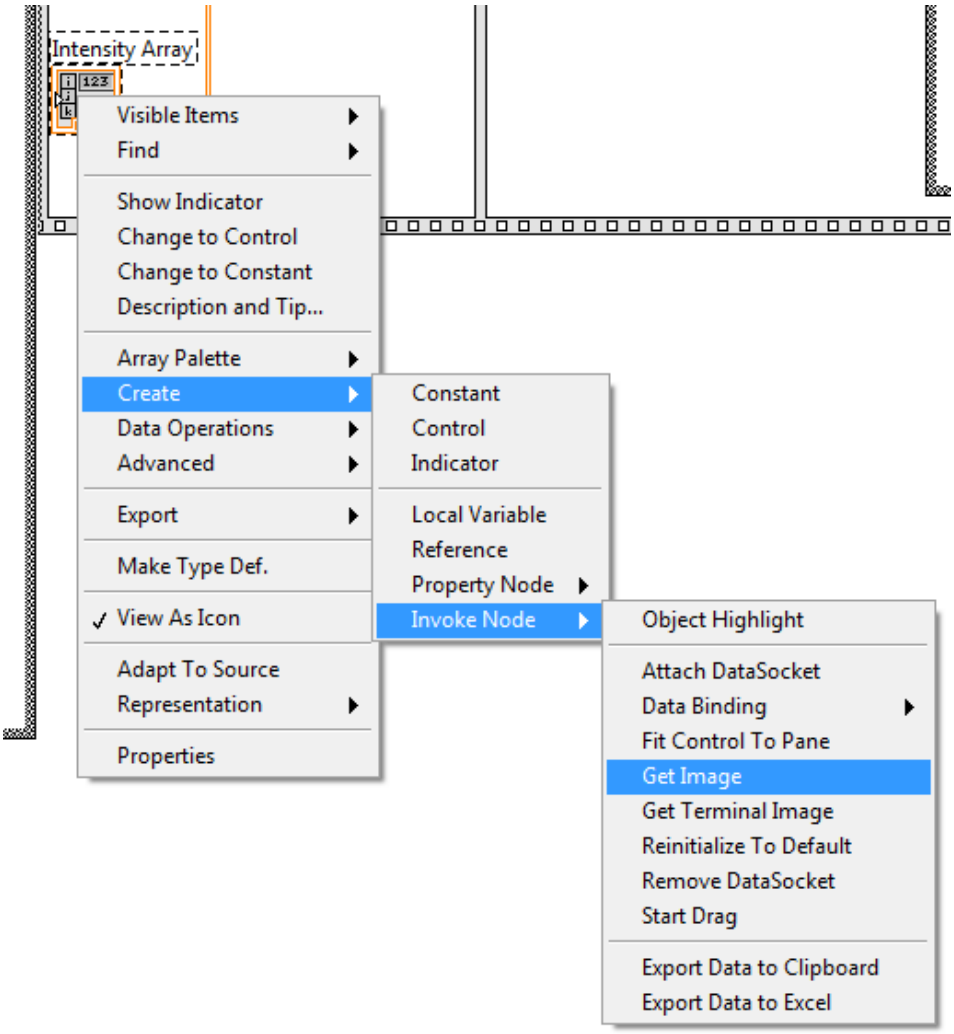
On a given day the lab may make dozens of measurements, the ability to quickly look at an image of the measurement is extremely useful for tracking down significant measurements. I figured out how to take a picture of the chart or 2D map of a measurement exactly how it appears in LabVIEW and save the image for later viewing, allowing an individual to very rapidly sort through images. I started then working this into all the measurement programs used in the lab.



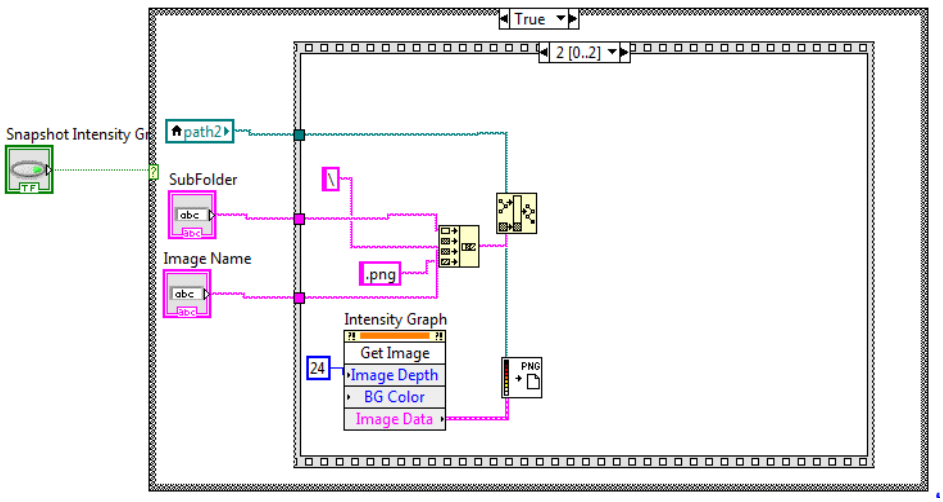

*Example of image folder containing that day's scans, this allows a full day's worth of work to be perused in seconds instead of the typical 15-20 minutes that it takes to open and graph each file individually.*

As simple step by step instructions on how to retrieve graph images was requested, I have provided them here.

1. Locate the graph's terminal
2. Right click on the graph
    a. Navigate to Create->Invoke Node->Get Image



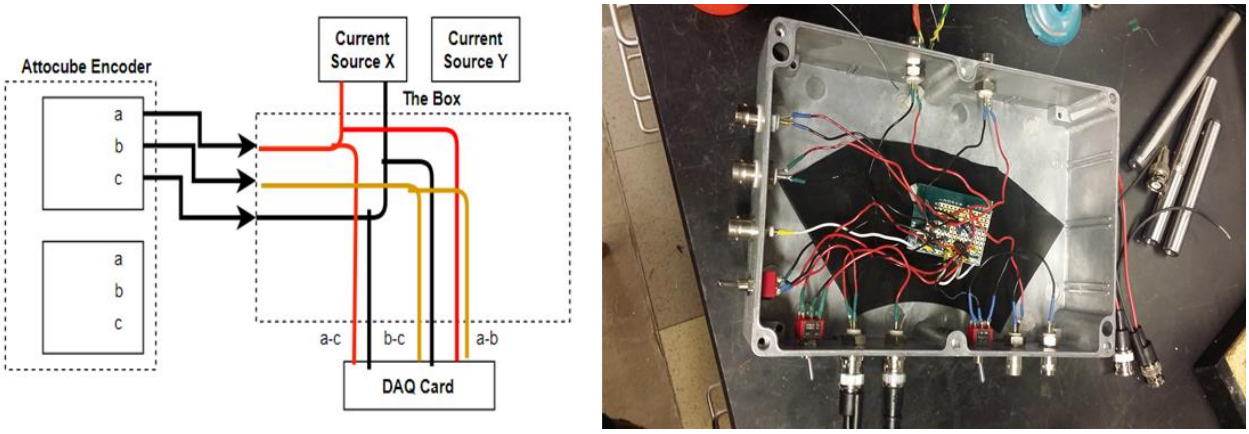

4. This will give you a construct that will output the image data as a matrix, this can be converted to an image file using a “to png file” construct and appropriate file path.




*Example LabVIEW structure that saves an image of a graph*

## 4.4 Encoder Box

The Attocube positioning system that the experimental setup uses contains a set of encoders for determining the coarse position. This project involved developing a system of resistance measurement using two current sources and a DAQ card. An enclosure was machined to have the appropriate BNC connectors and switches, this arrangement was the soldered together and a program was written to measure the voltage data.

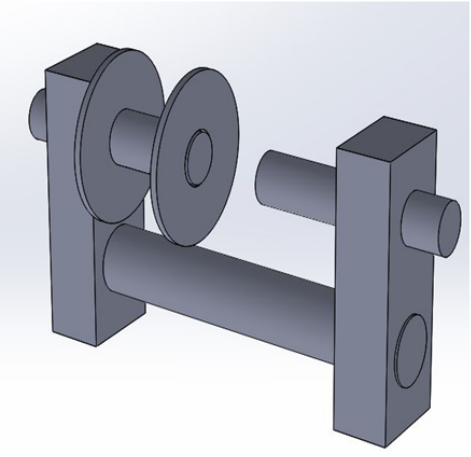


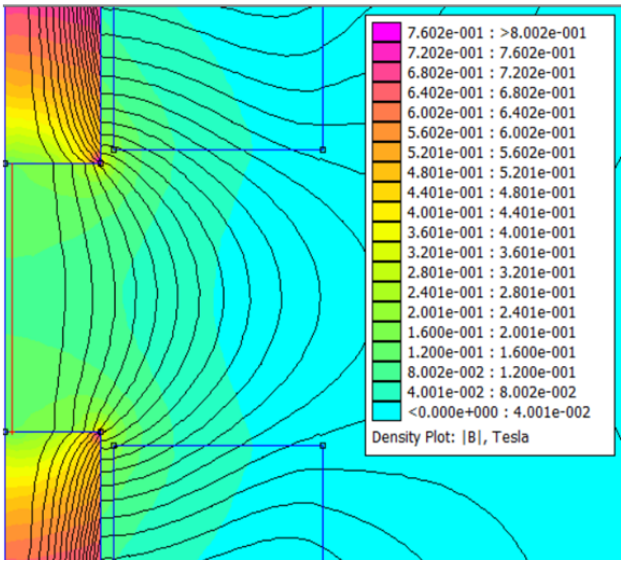

*On the left: wiring diagram for measurement, on the right: physical switch box with wiring exposed*

In addition to the wiring for this setup, I also developed a LabVIEW program to use it the system.

## 4.5 Magnet Design Project

For this project I was asked to design a replacement for an electromagnet being used on an experimental setup in lab. This was to be a C magnet, I built solidworks models for the design and researched how to simulate the field intensity with different applied currents.






*On the left: Solidworks model of magnet, design consists of spool of wire mounted on paramagnetic material. On the right: FEMM model showing expected field strength of design.*

# Appendix

## A1: Code for fitting multiple waves to a single spectroscopy wave, written for Igor pro

```
#pragma rtGlobals=3           // Use modern global access method and strict
wave access.

// This function gives you control over the variables from a single point
function declareVars()
       variable/G a1, a2, a3, b1, b2, b3, f1, f2, f3, offset, startPoint,
endPoint, f1Window, f2Window, f3Window, cutVal1, cutVal2, cutVal3, b1Window,
b2Window, b3Window
       offset = 1
       a1 = 1
       a2 = 100
       a3 = 300
       b1 = 1
       b2 = 6
       b3 = 6
       f1 = 10
       f2 = 615.5
       f3 = 631

      // Distance Either side of point that it will fit
       f1Window =1
       f2Window = 1
       f3Window  = 0.5

       b1Window = 1
       b2Window = 2
       b3Window = 1

      // Point at which to zero out values
      cutVal1 = 1700
      cutVal2 = 1000
      cutVal3 = 2500

      // array value at which to start and stop fitting curves
      startPoint = 20
      endPoint = 1000

end

function cutVals(a, b)
      Variable a; Variable b
      if(a > b)
            return 0
      endif
      return a
```

```
end


//This function automatically loads all files in a folder, grabs the x and y
values from the files then tries to fit
function fitFolder()
	string wname,fname,xname,yname, xMaxString, yMaxString, xMinString,
yMinString, firstFileName, lastFileName, xStepString, yStepString

	//Ask the user to identify a folder on the computer
	getfilefolderinfo/D


	//=========================
	// Function coefficients etc
	//=========================
	 variable/G a1, a2, a3, b1, b2, b3, f1, f2, f3, offset, startPoint,
endPoint, f1Window, f2Window, f3Window, cutVal1, cutVal2, cutVal3, b1Window,
b2Window, b3Window
	 offset = 939
	 a1 = 1200
	 a2 = 800
	 a3 = 1000
	 b1 = 3.2
	 b2 = 5.5
	 b3 = 10
	 f1 = 606.76
	 f2 = 620.13
	 f3 = 632.82

	// Distance Either side of point that it will fit
	 f1Window = 3
	 f2Window = 5
	 f3Window  = 5

	 b1Window = 1
	 b2Window = 3
	 b3Window = 1

	// Point at which to zero out values
	cutVal1 = 1700
	cutVal2 = 1200
	cutVal3 = 1200

	// array value at which to start and stop fitting curves
	startPoint = 75
	endPoint = 500


	declareVars()

	string holdString = "0100100100" // coefficients to hold

	//Store the folder that the user has selected as a new symbolic path in
IGOR called cgms
	newpath/O cgms S_path
```

```
	//Create a list of all files that are .txt files in the folder. -1
parameter addresses all files. "????" means alll extensions
	string filelist= indexedfile(cgms,-1,"????")



	//=============================================
	//     Find the step size bounds and axis for the graph
	//=============================================
	firstFileName = stringfromlist(0,filelist) // First
	lastFileName = stringfromlist((itemsinlist(filelist)-1),filelist) //
Last

	// y value and x value regular expressions
	string xmatch = "x= ([[:digit:]]+)"
	string ymatch = "y= ([[:digit:]]+)"
	// Get the max and min x/y values from first and last file
	SplitString/E=(xmatch) firstFileName, xMinString
	SplitString/E=(xmatch) lastFileName, xMaxString
	SplitString/E=(ymatch) firstFileName, yMinString
	SplitString/E=(ymatch) lastFileName, yMaxString
	variable xMin = str2num(xMinString)
	variable xMax = str2num(xMaxString)
	variable yMin = str2num(yMinString)
	variable yMax = str2num(yMaxString)
	// Get the x step size and x length
	lastFileName = stringfromlist(1,filelist) // file one step forward
	SplitString/E=(xmatch) lastFileName, xStepString
	variable xStepSize = str2num(xStepString) - xMin
	 // Calculate number of x steps
	variable numXIndices = ((xMax - xMin) / xStepSize) + 1

	// Get the y step size and y length
	lastFileName = stringfromlist(numXIndices,filelist) // file one row
forward
	SplitString/E=(ymatch) lastFileName, yStepString
	variable yStepSize = str2num(yStepString) - yMin
	variable numYIndices = ((yMax - yMin) / yStepSize) + 1


	//=============================================
	// Create and process graph
	//=============================================

	// Coefficients for the individual waves
	Make/D/N=10/O W_coef
	Make/D/N=10/O W_sigma

	// Create Peak Maps
	Make/D/N=(numXIndices,numYIndices)/O peakMap1
	Make/D/N=(numXIndices,numYIndices)/O peakMap2
	Make/D/N=(numXIndices,numYIndices)/O peakMap3
	Make/D/N=(numXIndices,numYIndices)/O errorMap
	// Set x and y scaling
	SetScale/I x xMin,xMax,"", peakMap1
	SetScale/I y yMin,yMax,"", peakMap1
	SetScale/I x xMin,xMax,"", peakMap2
```

```
	SetScale/I y yMin,yMax,"", peakMap2
	SetScale/I x xMin,xMax,"", peakMap3
	SetScale/I y yMin,yMax,"", peakMap3
	SetScale/I x xMin,xMax,"", errorMap
	SetScale/I y yMin,yMax,"", errorMap


	// Loop indices
	variable xIndex = 0
	variable yIndex = 0
	variable i = 0
	do
		fname = stringfromlist(i,filelist)
		wname = "original " + fname[0,7] + "-"
		LoadWave/G/D/A=$wname/P=cgms/O/L={0,0,0,0,0}
stringfromlist(i,filelist)

		xname =  "original " + fname[0,7] + "-0"
		yname =  "original " + fname[0,7] + "-1"


		W_coef[0] = {offset,a1,a2,a3,b1,b2,b3,f1,f2,f3} // assign
coefficients


		// Create constraints
		Make/O/T/N=6 T_Constraints
		//T_Constraints[0] = {"K1 > 0.01","K2 > 0.01","K3 > 0.01","K4 >
0.01","K5 > 0.01","K6 > 0.01"}
		T_Constraints[0] = {"K0 < 50", "K0 > 0.05", "K2 > 0.05","K3 >
0.05", "K8 > " + num2str(f2 - f2Window) ,"K8 < " + num2str(f2 + f2Window),"K9
> " + num2str(f3 - f3Window) ,"K9 < " + num2str(f3 + f3Window), "K5 < " +
num2str(b2 + b2Window), "K5 > " + num2str(b2 - b2Window), "K6 < " +
num2str(b3 + b3Window), "K6 > " + num2str(b3 - b3Window), "K3 < 700"}

		// Fit to curve
		FuncFit/Q/H=(holdString)/TBOX=1016 F W_coef
$yname[startPoint,endPoint]  /X=$xname /D/C=T_Constraints


		// Record coefficients a1, a2, a3
		errorMap[xIndex][yIndex] = abs(W_sigma[1]) + abs(W_sigma[2]) +
abs(W_sigma[3]) +  abs(W_sigma[4]) + abs(W_sigma[5]) + abs(W_sigma[6])
		if(errorMap[xIndex][yIndex] < 2500000)
			peakMap1[xIndex][yIndex] = cutVals(W_coef[1], cutVal1)
			peakMap2[xIndex][yIndex] = cutVals(W_coef[2], cutVal2)
			peakMap3[xIndex][yIndex] = cutVals(W_coef[3], cutVal3)
		else
			peakMap1[xIndex][yIndex] = 0
			peakMap2[xIndex][yIndex] = 0
			peakMap3[xIndex][yIndex] = 0
		endif
		// Move to next position in matrix
		xIndex += 1
		if(xIndex == numXIndices)
			xIndex = 0
			yIndex += 1
```

```
                  if(yIndex == numYIndices)
                        yIndex = 0
                  endif
            endif

            //. Clear waves
            KillWaves $xname
            KillWaves $yname
            KillWaves $("fit_" + yname)

            i += 1            //move to next file
      //while(i<100)
      while(i<itemsinlist(filelist))                  //end when all files
are processed.

      // Uncomment to create graphs
      //Display
      //AppendImage peakMap1
      //SetWindow kwTopWin,hook(myHook)=peakMapHook
      //Display
      //AppendImage peakMap2
      //SetWindow kwTopWin,hook(myHook)=peakMapHook
      //Display
      //AppendImage peakMap3
      //SetWindow kwTopWin,hook(myHook)=peakMapHook

end

Function F(w, x) : FitFunc
      WAVE w; Variable x

      return  w[0]+ w[1]*(w[4]^2)/((x-w[7])^2 + w[4]^2)*0 +
w[2]*(w[5]^2)/((x-w[8])^2 + w[5]^2) +w[3]*(w[6]^2)/((x-w[9])^2 + w[6]^2)
End


Function getFit(x, y)
      Variable x; Variable y

      string wname,fname,xname,yname, xMaxString, yMaxString, xMinString,
yMinString, firstFileName, lastFileName, xStepString, yStepString

      //Ask the user to identify a folder on the computer
      getfilefolderinfo/D


      //=========================
      // Function coefficients etc
      //=========================
       variable/G a1, a2, a3, b1, b2, b3, f1, f2, f3, offset, startPoint,
endPoint, f1Window, f2Window, f3Window, cutVal1, cutVal2, cutVal3, b1Window,
b2Window, b3Window
       offset = 939
       a1 = 1200
       a2 = 800
       a3 = 1000
       b1 = 3.2
```

```
	b2 = 5.5
	b3 = 10
	f1 = 606.76
	f2 = 620.13
	f3 = 632.82

	// Distance Either side of point that it will fit
	f1Window = 3
	f2Window = 5
	f3Window  = 5

	b1Window = 2
	b2Window = 2
	b3Window = 2

	// Point at which to zero out values
	cutVal1 = 1700
	cutVal2 = 1200
	cutVal3 = 1200

	// array value at which to start and stop fitting curves
	startPoint = 75
	endPoint = 500

	declareVars()


	string holdString = "0100100100" // coefficients to hold

	//Store the folder that the user has selected as a new symbolic path in
IGOR called cgms
	newpath/O cgms S_path

	//Create a list of all files that are .txt files in the folder. -1
parameter addresses all files. "????" means alll extensions
	string filelist= indexedfile(cgms,-1,"????")



	//=============================================
	//     Find the step size bounds and axis for the graph
	//=============================================
	firstFileName = stringfromlist(0,filelist) // First
	lastFileName = stringfromlist((itemsinlist(filelist)-1),filelist) //
Last

	// y value and x value regular expressions
	string xmatch = "x= ([[:digit:]]+)"
	string ymatch = "y= ([[:digit:]]+)"
	// Get the max and min x/y values from first and last file
	SplitString/E=(xmatch) firstFileName, xMinString
	SplitString/E=(xmatch) lastFileName, xMaxString
	SplitString/E=(ymatch) firstFileName, yMinString
	SplitString/E=(ymatch) lastFileName, yMaxString
	variable xMin = str2num(xMinString)
	variable xMax = str2num(xMaxString)
	variable yMin = str2num(yMinString)
```

```
	variable yMax = str2num(yMaxString)
	// Get the x step size and x length
	lastFileName = stringfromlist(1,filelist) // file one step forward
	SplitString/E=(xmatch) lastFileName, xStepString
	variable xStepSize = str2num(xStepString) - xMin
	 // Calculate number of x steps
	variable numXIndices = ((xMax - xMin) / xStepSize) + 1

	// Get the y step size and y length
	lastFileName = stringfromlist(numXIndices,filelist) // file one row
forward
	SplitString/E=(ymatch) lastFileName, yStepString
	variable yStepSize = str2num(yStepString) - yMin
	variable numYIndices = ((yMax - yMin) / yStepSize) + 1

	Make/D/N=10/O W_coef
	Make/D/N=10/O W_sigma

	string xString, yString
	variable xIndex = 0
	variable yIndex = 0
	variable i = 0
	SplitString/E=(xmatch) stringfromlist(i,filelist), xString
	xIndex = str2num(xString)
	SplitString/E=(ymatch) stringfromlist(i,filelist), yString
	yIndex = str2num(yString)


	// Go through files until the position is found
	do
		i += 1
		SplitString/E=(xmatch) stringfromlist(i,filelist), xString
		xIndex = str2num(xString)
		SplitString/E=(ymatch) stringfromlist(i,filelist), yString
		yIndex = str2num(yString)

	while(yIndex != y || xIndex != x)

		print("i= " + num2str(i))
		print("x= " + num2str(xIndex))
		print("y= " + num2str(yIndex))

		fname = stringfromlist(i,filelist)
		wname = fname[0,16] + "-"
		LoadWave/G/D/A=$wname/P=cgms/O/L={0,0,0,0,0}
stringfromlist(i,filelist)

		xname =  fname[0,16] + "-0"
		yname =  fname[0,16] + "-1"


		W_coef[0] = {offset,a1,a2,a3,b1,b2,b3,f1,f2,f3} // assign
coefficients

		// Create constraints
		Make/O/T/N=6 T_Constraints
```

```
            //T_Constraints[0] = {"K1 > 0.01","K2 > 0.01","K3 > 0.01","K4 >
0.01","K5 > 0.01","K6 > 0.01"}
            T_Constraints[0] = {"K0 < 50", "K0 > 0.05", "K2 > 0.05","K3 >
0.05", "K8 > " + num2str(f2 - f2Window) ,"K8 < " + num2str(f2 + f2Window),"K9
> " + num2str(f3 - f3Window) ,"K9 < " + num2str(f3 + f3Window), "K5 < " +
num2str(b2 + b2Window), "K5 > " + num2str(b2 - b2Window), "K6 < " +
num2str(b3 + b3Window), "K6 > " + num2str(b3 - b3Window), "K3 < 700"}

            // Fit to curve
            FuncFit/H=(holdString)/TBOX=1016 F W_coef
$yname[startPoint,endPoint]  /X=$xname /D/C=T_Constraints
            string graphName = "coords: (" + num2str(x) + ", " + num2str(y) +
")"
            string windowName = "FitAtPoint"

            // Create individual peaks
            Duplicate/O $xname, peak1
            Duplicate/O $xname, peak2
            Duplicate/O $xname, peak3
            // Apply functions to individual peaks
            peak1 = W_coef[0]+ W_coef[1]*(W_coef[4]^2)/((peak1-W_coef[7])^2 +
W_coef[4]^2)
            peak2 = W_coef[0]+ W_coef[2]*(W_coef[5]^2)/((peak2-W_coef[8])^2 +
W_coef[5]^2)
            peak3 = W_coef[0]+ W_coef[3]*(W_coef[6]^2)/((peak3-W_coef[9])^2 +
W_coef[6]^2)


            Display
            AppendToGraph $yname vs $xname
            // C = R, G, B
            AppendToGraph/C=(2000,35035,2000) peak1 vs $xname
            AppendToGraph/C=(65035,35035,2000) peak2 vs $xname
            AppendToGraph/C=(45035,15035,15035) peak3 vs $xname

            string newName = "fit_" + yname
            AppendToGraph/C=(2000,2000,65035) $newName

            TextBox/A=RT ""
            // Add the coefficients to graph
            AppendText(graphName)
            AppendText("Coefficients")
            AppendText( "a1 = " + num2str(W_coef[1]) + " ± " +
num2str(W_sigma[1]) )
            AppendText( "a2 = " + num2str(W_coef[2]) + " ± " +
num2str(W_sigma[2]) )
            AppendText( "a3 = " + num2str(W_coef[3]) + " ± " +
num2str(W_sigma[3]) )
            AppendText( "b1 = " + num2str(W_coef[4]) + " ± " +
num2str(W_sigma[4]) )
            AppendText( "b2 = " + num2str(W_coef[5]) + " ± " +
num2str(W_sigma[5]) )
            AppendText( "b3 = " + num2str(W_coef[6]) + " ± " +
num2str(W_sigma[6]) )
            AppendText( "f1 = " + num2str(W_coef[7]) + " ± " +
num2str(W_sigma[7]) )
```

```
            AppendText( "f2 = " + num2str(W_coef[8]) + " ± " +
num2str(W_sigma[8]) )
            AppendText( "f3 = " + num2str(W_coef[9]) + " ± " +
num2str(W_sigma[9]) )


End



Function peakMapHook(s)
      STRUCT WMWinHookStruct &s

      Variable hookResult = 0

      switch(s.eventCode)
            case 0:                        // Activate
                  // Handle activate
                  break

            case 1:                        // Deactivate
                  // Handle deactivate
                  break

            case 5:                        // Mouse down
                  print("click")
                  variable xPixel = s.mouseLoc.v
                  variable yPixel = s.mouseLoc.h
                  print(TraceFromPixel(xPixel, yPixel, ""))
                  break
            // And so on . . .
      endswitch

      return hookResult       // 0 if nothing done, else 1
End
```

# References

All figures original unless another reference has been provided.

[1] E. J. McCormick, M. J. Newburger, Y. K. Luo, K. M. McCreary, S. Singh, I. B. Martin, E. J. Cichewicz, B. T. Jonker, and R. K. Kawakami, 2D Mater. **5**, 011010 (2017).

[2] P. J. Zomer, M. H. D. Guimarães, N. Tombros, and B. J. van Wees. Long-distance spin transport in high-mobility graphene on hexagonal boron nitride. Phys. Rev. B, 86:161416, Oct 2012.